UUID/GUID White Paper

Version 1.1

November 16, 1998
Intel Corporation

TABLE OF CONTENTS

LIST OF TABLES

Purpose and Scope

The purpose of this document is to bring together information about the Universally Unique Identifier/Globally Unique Identifier (UUID/GUID) that will help designers understand the UUID/GUID requirements and assist with their implementation. Much of this document has been copied from the Wired for Management (WfM) Baseline 1.1a specification, the System Management BIOS (SMBIOS) 2.x specifications and other reference documents. This white paper should supplement the relevant specifications, not replace them. Therefore, this document can be used as a guide to aid implementation of the UUID/GUID, along with the other referenced documents.

Revision History

Revision	Date	Comments
1.1	11/16/98	Updates to spec versions, names and URLs
1.0	7/01/98	Initial Release

Introduction

The UUID, also known as GUID, is a 128-bit number intended to uniquely identify a system on a network. This number must remain the same, despite hardware and software configuration changes.

One of the main reasons for using UUIDs is that no centralized authority is required to administer them (beyond the one that allocates IEEE 802.1 node identifiers [MAC address]). As a result, generation-on-demand can be completely automated, and UUIDs can be used for a wide variety of purposes.

To support this identification, the BIOS must provide a location in memory for the UUID and methods of access. The UUID must be programmed into non-volatile memory on the motherboard so that a change of motherboard would be the only instance when this number would change.

UUID/GUID Generation

Appendix J of the Wired for Management Baseline Specification Version 1.1a provides algorithms guaranteed to generate UUIDs either different from all others generated until 3400 A.D. or extremely likely to be different (depending on the mechanism implemented). One algorithm uses the MAC (IEEE 802) address to generate a UUID. This method can be used for systems that have integrated LAN. A second algorithm is defined that can be used to generate a UUID for systems that do not have integrated LAN.

UUID/GUID Programming

The UUID must be programmed into non-volatile memory on the motherboard during system/motherboard manufacture. The system BIOS is not responsible for generating the UUID. The method for permanently writing and storing a UUID or any other SYSID is defined in the SYSID Programming Interface Specification.

Design Considerations

The UUID must be protected from BIOS upgrades; that is, BIOS upgrades must not reset or clear the UUID. The UUID must remain the same for the life of the motherboard. After accessing the UUID non-volatile memory location, the BIOS must use the UUID to construct the SMBIOS Tables and the SYSID Tables (if SYSID is supported).

If the UUID is initially programmed to all ZEROS, it is not present and not programmable. This scenario should be avoided to allow generation and permanent programming of the UUID, and to adhere to the Draft Wired for Management 2.0 specification, which requires that the "UUID field identifies the system's NON-ZERO UUID value." This also enables backward-compatibility with the Wired for Management 1.1a Baseline, which requires a SYSID Structure and a "PXENV unique system ID structure."

If the UUID is initially programmed to all ONES (all FFh) to enable future programming, a valid UUID must be programmed into non-volatile memory before shipping the motherboard. Failure to do so will result in a system that lacks the required valid UUID as defined by the Wired for Management Baseline specification v2.0.

Design Approaches

The two approaches for ensuring that a valid UUID exists in the system before shipping are:

1. Program the UUID into non-volatile memory during the manufacture of the motherboard. UUID Generation may utilize the Random number or the IEEE 802 Address approach. Implementing the IEEE 802 Address method requires knowledge of the LAN On Motherboard or the Network Interface Card MAC address.

2. Initially program the UUID to all ONES (FFh) during manufacture of the motherboard and then program a valid UUID before shipping the system/motherboard. As previously mentioned, the UUID must reside in protected non-volatile memory. Depending on the presence of LAN on Motherboard or a Network Interface Card, this approach may provide the developer with the option of generating the UUID based on an IEEE 802 address rather than a random number.

BIOS Implementations

The following sections outline the Desktop Management BIOS (DMBIOS) and System Management BIOS (SMBIOS) structures for accessing the UUID.

DMBIOS 2.0

A DMBIOS 2.0 BIOS must use table-based SYSID structures that consist of the SYSID Entry Point Structure Table and the SYSID BIOS Structure Table.

The interface requirements for SYSID BIOS support are :

- System BIOS must provide full Desktop Management Interface (DMI) 2.0 support.
- System BIOS must provide SYSID capability. One way of doing this is to provide the UUID/GUID capability at the motherboard level. This provides a mechanism for the storage of a UUID without the BIOS being required to actually generate the UUID.

The SYSID is a superset of identifiers that allows definition and organization of unique identifiers. The superset is comprised of identifiers such as UUID and 1394 ID. It is important to understand that a UUID is a SYSID, but a SYSID need not be a UUID.

The SYSID Entry Point Structure Table provides a pointer to the address of the beginning of the SYSID BIOS Structure Table (byte aligned) and lists the total number of structures within the SYSID BIOS Structure Table. Currently there are two defined and supported SYSID BIOS Structures, UUID and 1394 ID.

For details on programming the SYSID values, refer to the SYSID Programming Interface Specification Version 1.2.

SYSID Entry Point Structure—This will be found in the 000E0000h to 000FFFFFh physical address area of memory/RAM. **Important: The Entry Point Structure is PARAGRAPH Aligned.**

Table 1 - SYSID Entry Point Structure Format

ELEMENT	LENGTH (in bytes)	DESCRIPTION
Header/Type	7	_SYSID_
Checksum	1	Checksum of the SYSID BIOS Entry Point Structure
Length	2	Total length of SYSID BIOS Structure Table (set to 011h)
SYSID BIOS Structure Table Address	4	32-bit physical address of the beginning of the SYSID BIOS Structure Table. **Important: This value is** BYTE Aligned!!
Number of SYSID BIOS Structures	2	Total number of structures within the SYSID BIOS Structure Table
SYSID BIOS Revision	1	Revision of the SYSID BIOS Extensions (set to 00h)

SYSID BIOS Structure Table

The UUID Structure Format contains the actual UUID data which should be obtained from non-volatile memory.

Table 2 - SYSID Structure (UUID Type) Format

ELEMENT	LENGTH (in bytes)	DESCRIPTION
Header/Type	6	_UUID_
Checksum	1	Checksum of the UUID BIOS Structure
Length	2	Total length of UUID BIOS Structure (set to 0019h)
Variable Data Portion	16	Actual UUID data (obtained from non-volatile memory)

SMBIOS 2.1

There are two access methods defined for the SMBIOS structures; one or both methods can be used in an SMBIOS v2.1-compliant BIOS.

- **Table Method:** As defined in the SMBIOS v2.1 Specification, the Table Convention uses a "searchable entry-point structure which contains a pointer to the packed SMBIOS structures residing somewhere in 32-bit physical address space." This access method provides a means of accessing the SMBIOS structures under 32-bit protected-mode operating systems.

- **Plug-and-Play (PnP) Calling Method:** This access method allows using Plug-and-Play BIOS functions to access DMI information. Plug and Play functions 50h-5Fh access the SMBIOS Tables and are available in real-mode and 16-bit protected-mode.

Note: SMBIOS implementers may choose to implement one or both access methods. However, some DMI browsers that rely on the Plug-and-Play Calling Convention may no longer work if only the table-based method is implemented.

Table 3 - System Information (Type 1) Structure Format

Offset	Spec Version	Name	Length	Value	Description
00h	2.0+	Type	BYTE	1	Component ID Information Indicator
01h	2.0+	Length	BYTE	08h or 19h	Length dependent on version supported, 08h for 2.0 or 19h for 2.1
02h	2.0+	Handle	WORD	Varies	
04h	2.0+	Manufacturer	BYTE	STRING	Number of Null terminated string
05h	2.0+	Product Name	BYTE	STRING	Number of Null terminated string
06h	2.0+	Version	BYTE	STRING	Number of Null terminated string
07h	2.0+	Serial Number	BYTE	STRING	Number of Null terminated string
08h	2.1+	GUID	16 BYTEs	Varies	Globally Unique ID number. If the value is all FFh, the ID is not currently present in the system, but is settable. If the value is all 00h, the ID is not present in the system and not settable.
18h	2.1+	Wake-up Type	BYTE	ENUM	Identifies the event that caused the system to power up.

Table 4 - SMBIOS Entry Point Structure Format

Offset	Name	Length	Description
00h	Anchor String	4 BYTEs	_SM_, specified as four ASCII characters (5F 53 4D 5F)
04h	Entry Point Structure Checksum	BYTE	Checksum of the Entry Point Structure (EPS). Values are summed starting at offset 00h.
05h	Entry Point Length	BYTE	Length of the EPS, starting with the Anchor String field, in bytes.
06h	SMBIOS Major Version	BYTE	Identifies the major version of this specification implemented in the table structures.
07h	SMBIOS Minor Version	BYTE	Identifies the minor version of this specification implemented in the table structures.
08h	Maximum Structure Size	WORD	
0Ah		BYTE	
0Bh-0Fh	Reserved	5 BYTEs	Reserved for future assignment by this specification, must be set to all 00h.
10h	Intermediate Anchor String	5 BYTEs	_DMI_, specified as five ASCII characters (5F 44 4D 49 5F). **Note:** This field is paragraph-aligned, to allow legacy DMI browsers to find this entry point within the SMBIOS Entry Point Structure.
15h	Intermediate Checksum	BYTE	Checksum of Intermediate Entry Point Structure (IEPS). Values are summed starting at offset 10h, for 0Fh bytes.
16h	Structure Table Length	WORD	Total length of SMBIOS Structure Table, pointed to by the Structure Table Address, in bytes.
18h	Structure Table Address	DWORD	The 32-bit physical address of the read-only SMBIOS Structure Table, which can start at any 32-bit address. This area contains all of the SMBIOS structures fully packed together. These structures can be parsed to produce exactly the same format as that returned from a GET SMBIOS Structure function call.
1Ch	Number of SMBIOS Structures	WORD	Total number of structures present in the SMBIOS Structure Table. This is the value returned as NumStructures from the GET SMBIOS Information function call.
1Dh	SMBIOS BCD Revision	BYTE	Indicates compliance with a revision of this specification. For revision 2.1, the returned value is 21h. If the value is 00h, only the Major and Minor Versions in offsets 6 and 7 of the Entry Point Structure provide the version information.

SMBIOS 2.2/ SMBIOS 2.3

A BIOS compliant with the SMBIOS 2.2/2.3 Specification must provide the table-based access convention and can optionally provide the Plug-and-Play function interface.

BIOS Implementation Considerations

SYSID vs. SMBIOS Support

The current Wired for Management Baseline Specification (Version1.1a) requires the PXENV unique system ID structure " _SYSID_ "to be contained in the client's main memory when execution of the download bootstrap begins. There are no requirements for implementation of SMBIOS structures in the WfM Baseline Specification Version 1.1a.

The WfM Baseline specification v2.0 requires implementation of SMBIOS Version 2.2 or higher. This **requires** support for table-based SMBIOS structures and optionally allow for providing the SMBIOS Plug-and-Play function interface. The WfM Baseline specification also **recommends** support for the "_SYSID_" table-based structure.

When determining which method(s) to support in a BIOS implementation, carefully consider whether the design is also intended to meet the requirements of the PC98 and PC99 specifications.

Table 5 illustrates the required and recommended table-based structure support for the specifications mentioned in this section. This illustration shows that supporting both the SMBIOS table-based structure and the SYSID table-based structure is the best way to maintain compatibility with the PCxx specifications and the Wired for Management Baseline specification. Supporting both of the table-based methods and *additionally* supporting the SMBIOS PnP function interface will allow for backward-compatibility with existing DMI browsers.

Table 5 - Required and Recommended Table-Based Structure Support

	WfM Baseline 1.1a	WfM Baseline 2.0	PC98	PC99
SYSID	Required	Recommended	Required	Required
SMBIOS	N/A	Required (v2.2+)	N/A	Required* (v2.2)

* Required for Office and Workstation, recommended for mobile.

Note: SM BIOS v2.1 (or later) table support is recommended in the Intel/Microsoft Hardware Design Guide Version 2.0 for Microsoft Windows NT Server

Preboot Execution Environment (PXE) ROM Support

PXE-compliant boot ROMs **must** support UUID detection by reading table-based SMBIOS structures.

PXE-compliant boot ROMs **should** support UUID detection for legacy system support by:

- Reading SMBIOS structures through the PnP function interface.
- Reading table-based SYSID structures.

Reference Documents

Wired for Management Baseline Specification Version 1.1a
ftp://download.intel.com/ial/wfm/baseline.pdf

Wired for Management Baseline Specification Version 2.0
http://developer.intel.com/ial/wfm/wfmspecs.htm

Preboot Execution Environment (PXE) Specification Version 1.0
http://developer.intel.com/ial/wfm/wfmspecs.htm

Intel Preboot Execution Environment (PXE) PDK
http://developer.intel.com/ial/wfm/tools/pxe/index.htm

DMBIOS Interface Specification Version 2.0
http://www.ibm.com/products/surepath/other/smbios.html

SMBIOS Specification Version 2.1
http://www.ibm.com/products/surepath/other/smbios.html

SMBIOS Specification Version 2.2
http://www.ibm.com/products/surepath/other/smbios.html

SMBIOS Specification Version 2.3
ftp://download.intel.com/ial/wfm/smbios.pdf

Desktop Management Task Force, Inc. (DMTF) DMI Version 2.0 Specification
http://www.dmtf.org/tech/specs.html

SYSID BIOS Support Interface Requirements Version 1.2
http://developer.intel.com/ial/wfm/design/smbios/npnpspec.htm

SYSID Programming Interface Specification Version 1.2
http://developer.intel.com/ial/wfm/design/smbios/npnpspec.htm

Intel/Microsoft PC 98 System Design Guide Version 1.0
http://developer.intel.com/design/pc98/

Intel/Microsoft PC 99 System Design Guide Version 1.0
http://developer.intel.com/design/desguide/

Intel/Microsoft Hardware Design Guide Version 2.0 for Windows NT Server
http://developer.intel.com/design/servers/desguide/